****

**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **课程编号** |  | **课程名称** | **计算机网络** |
| **实验学分** |  | **任课教师** | **叶松涛** |
| **学生学号** | **190320020** | **行政班级** | **信科1901** |
| **学生姓名** | **田章** | **联系方式** | **17680314099** |

**20 22 -2022学年度2学期**

**教务处制**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | | | | |
| **实验编号** | **实验2** | **实验名称** | **UDP ping程序** | |
| **指导教师** | **叶松涛** | **实验日期** | **2022.5.20** | |
| **实验同组人** | **未分组** |  |  | |
| 1. **实验目的和要求**   目的：1.学习使用Python进行UDP套接字编程的基础知识。2.学习如何使用UDP套接字发送和接收数据报，以及如何设置适当的套接字超时。3.熟悉Ping应用程序及其在计算统计信息（如丢包率）中的作用。  要求：研究一个用Python编写的简单的ping服务器程序，并实现对应的客户端程序。这些程序提供的功能类似于现代操作系统中可用的标准ping程序功能。 | | | | |
| 1. **实验设备、软件**   笔记本+VScode（进行代码编辑和一些终端的命令操作）  Python环境：3.10.1 | | | | |
| 1. **实验内容（步骤、计划、方案）**   这个实验比较简单，我们直接开始编写服务端对应的客户端程序。  **主机名与端口号：**  由于服务端和客户端都是在本机上且服务端端口号是12000，因此我们将客户端ip地址设置为127.0.0.1，端口号设置为12000。  **消息格式**  客户端一共需要向服务端发送10条消息，因此我们用循环for i in range(10);发送的消息格式为Ping sequence\_number time,因此我们需要导入time模块来记录当前时间戳time1=time.time()，然后设置outputdata=f”Ping {i} {time1}”。  设置超时时间为1s后将消息头编码发送，如果发送成功，即没有超时我们则先接收消息，这里设置的接收消息缓存区大小为2048.然后对接收的消息进行解码，将解码后的信息加上记录时间的往返时延RTT信息进行打印输出；如果发送失败，则直接打印”输出超时” | | | | |
| 1. **实验原始记录、结果分析（可以附页）**   **UDPPingerServer.py(题目已给出)**  # UDPPingerServer.py  # We will need the following module to generate randomized lost packets  from socket import \*  import random  # Create a UDP socket  # Notice the use of SOCK\_DGRAM for UDP packets  serverSocket = socket(AF\_INET, SOCK\_DGRAM)  # Assign IP address and port number to socket  serverSocket.bind(('', 12000))  while True:      # Generate random number in the range of 0 to 10      rand = random.randint(0, 10)      # Receive the client packet along with the address it is coming from      message, address = serverSocket.recvfrom(1024)      # Capitalize the message from the client      message = message.upper()      # If rand is less is than 4, we consider the packet lost and do not respond      if rand < 4:          continue      # Otherwise, the server responds      serverSocket.sendto(message, address)  **UDPPingerClient.py**  # UDPPingerClient.py  from socket import \*  import time  serverName = '127.0.0.1'  serverPort = 12000  clientSocket = socket(AF\_INET, SOCK\_DGRAM)  for i in range(10):      time1 = time.time()      outputdata = f"Ping {i} {time1}"      # 设置超时 单位秒      clientSocket.settimeout(1)      clientSocket.sendto(outputdata.encode(), (serverName, serverPort))      try:          modifiedMessage, serverAddress = clientSocket.recvfrom(2048)          timeDiff = time.time() - time1          print(f"{modifiedMessage.decode()} RTT:{int(timeDiff\*1000)}ms")      except:          print("请求超时")  **在VScode中运行UDPPingerServer.py，然后在VScode终端中拆分终端，在右边再开一个终端输入python UDPPingerClient.py运行客户端代码。** | | | | |
| 1. **实验思考题及其解答**   **练习1目前，程序计算每个数据包的往返时间（RTT），并单独打印出来。请按照标准ping程序的模式修改。您需要在客户端每次ping后显示最小，最大和平均RTT。另外，还需计算丢包率（百分比）。**  **思考：我们需要先了解标准ping的输出格式以及超时的默认时长来判别怎样才算超时，也即ping失败**  在VScode中打开终端，我们先看一下Ping命令的标准格式是怎样的，因此在终端中输入Ping [www.baidu.com](http://www.baidu.com)    ping命令不带参数的话默认是ping 4次，这里四次均是能够ping成功的，为了知道失败的输出效果，我们直接ping一下谷歌    这里四次ping命令均是失败的，至此我们了解了成功与失败的标准Ping程序模式，那么失败即请求超时是怎么判定的呢？经过谷歌搜索了解到，不带参数的ping命令默认超时时间为5s。  **下面是程序的编写：**  **UDPPingerClient.py**  # UDPPingerClient.py  from socket import \*  import time  serverName = '127.0.0.1'  serverPort = 12000  clientSocket = socket(AF\_INET, SOCK\_DGRAM)  print(f"正在 Ping {serverName} 的数据:")  # 由于我们设置的超时值为5秒，这里可以将最小时差设置为6，这样第一次ping成功必然会刷新其值  receNum, maxDiffTime, minDiffTime, sumDiffTime = 0, -1, 6, 0  # 消息数为10  MESSAGE\_NUMBER = 10  for i in range(MESSAGE\_NUMBER):      time1 = time.time()      outputdata = f"Ping {i} {time1}"      # 设置超时 单位秒      clientSocket.settimeout(5)      clientSocket.sendto(outputdata.encode(), (serverName, serverPort))      try:          modifiedMessage, serverAddress = clientSocket.recvfrom(2048)          timeDiff = time.time() - time1          print(f"来自 {serverName} 的回复：字节={len(outputdata)} 时间={int(1000\*timeDiff)}ms")          receNum += 1          sumDiffTime += timeDiff          if timeDiff > maxDiffTime:              maxDiffTime = timeDiff          if timeDiff < minDiffTime:              minDiffTime = timeDiff      except:          print("请求超时。")  print(f"\n{serverName} 的 Ping 统计信息:")  print(f"\t数据包: 已发送 = {MESSAGE\_NUMBER}, 已接收 = {receNum}, 丢失 = {10-receNum} ({int((10-receNum)\*100/MESSAGE\_NUMBER)}% 丢失),")  if receNum != 0:      print("往返行程的估计时间(以毫秒为单位):")      print(f"\t最短 = {int(1000\*minDiffTime)}ms, 最长 = {int(1000\*maxDiffTime)}ms, 平均 = {int(1000\*sumDiffTime/receNum)}ms")  **在VScode中打开终端，输入**python UDPPingerClientStandard.py    需要说明的是，这里ping的服务器就是我们自己的主机，理论上来说应该是不存在丢包的，这里问题出在服务器是模拟的UDP，且存在40%的概率会丢包。  **练习2** UDP Ping**的另一个类似的应用是**UDP Heartbeat**。心跳可用于检查应用程序是否已启动并运行，并报告单向丢包。客户端在UDP数据包中将一个序列号和当前时间戳发送给正在监听客户端心跳的服务器。服务器收到数据包后，计算时差，报告丢包（若发生）。如果心跳数据包在指定的一段时间内丢失，我们可以假设客户端应用程序已经停止。实现**UDP Heartbeat**（客户端和服务器端）。您需要修改给定的**UDPPingerServer.p**y和您自己的**UDP ping**客户端。**  **思考：这题比较简单，只需要在服务器程序中设置一个死循环接收来自客户端的数据包，由于可能存在报告丢包，因此将接收数据包的语句放入try语句中，没有异就打印RTT，发生异常就打印”丢包”。客户端则每隔3秒就发送一次数据报。**  **代码如下：**  **UDPHeartbeatServer.py**  # UDPHeartbeatServer.py  # We will need the following module to generate randomized lost packets  import time  from socket import \*  # Create a UDP socket  # Notice the use of SOCK\_DGRAM for UDP packets  serverSocket = socket(AF\_INET, SOCK\_DGRAM)  # Assign IP address and port number to socket  serverSocket.bind(('', 12000))  while True:      try:          message, address = serverSocket.recvfrom(1024)          message = message.decode()          message = message.split()[1]          timeDiff = time.time() - float(message)          print(f"receive RTT: {1000\*timeDiff}ms")      except:          print("请求超时")  **UDPHeartbeatClient.py**  # UDPHeartbeatClient.py  from socket import \*  import time  serverName = '127.0.0.1'  serverPort = 12000  clientSocket = socket(AF\_INET, SOCK\_DGRAM)  while True:      time1 = time.time()      outputdata = 'Heartbeat ' + str(time1)      clientSocket.sendto(outputdata.encode(), (serverName, serverPort))  time.sleep(3)  **在VScode终端中输入**python UDPHeartbeatClient.py**,** | | | |
| 1. **实验评分**   **教师签名：**    **年 月 日** | | | |